ELBUG
SUPPLEMENT TO BEEBUG

VOLUME 3 NUMBER 6 MAY 1986

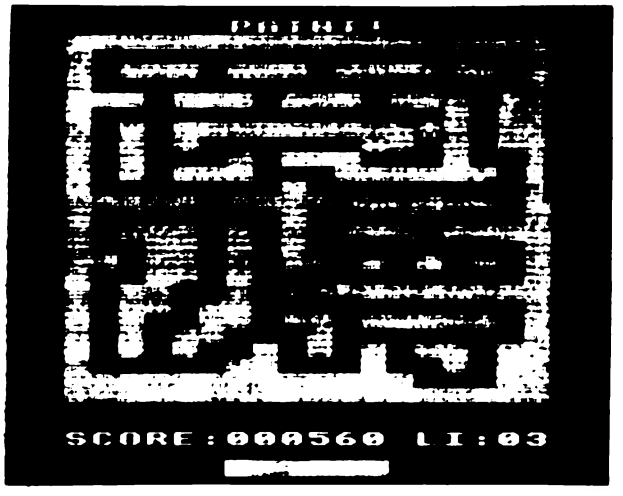
PAINTER

Avoid the monster and find your way around the maze in this fast moving, full colour game from Kevin Monro.

This fast game is simple to play but hard to win. You control the inevitable little man, who is trapped in a maze, with the usual Z and X keys for left and right and * and ? for up and down. To escape from the maze every inch of the floor must be trodden by you or the monster. Yes, there is the inevitable monster that inhabits the maze. Not only is the monster extremely nasty, but it also has a very keen sense of smell. It can always tell which direction it has to go in order to find you and it wastes no time in tracking you down.

The good news is that you don't have to travel over areas of the maze that the monster has covered. Once the whole maze has been covered, you must move back to your starting point (the top left hand corner) and press Return. The computer then checks that the whole maze has indeed been covered and then lets you free...into the next maze!

Be careful typing in the maze data in lines 1350-1530. A mistake here might mean that a section of a maze is shut off and not accessible to you or the monster. You can set up your own mazes by altering this data. Each section of 18 figures



G CASSETTE ELBUG CASSETTE ELBUG CASSETTE ELBUG CASSETTE ELBUG CASSETTE ELBUG CASSETTE E

This month's ELBUG magazine cassette includes several of the programs from this month's BEEBUG - the Mandelbrot set program, the Do-It-Yourself Basic program to add your own commands to Basic, the First Course examples on random numbers, the *SPOOL graphics utility, the Workshop procedures on stacks and queues, and the Ebony Castle game plus the Search and Replace utility and Painter game in this month's ELBUG supplement. All this for just £3.00 plus 50p post and packing.

Note that ELBUG cassettes from Vol.3 No.3 onwards are only available to order. Please address all orders to the St. Albans address, mark the envelope 'ELBUG cassette', and allow at least ten days for delivery.

represents one horizontal line of the maze. A one in any position gives a solid wall at that position and a zero gives a passage. Note that each section must start and end with ones and the first and last section must be all ones, to give the perimeter walls to the maze.

```
   10 REM PROGRAM PAINTER
   20 REM VERSION E0.1
   30 REM AUTHOR  KEVIN MUNRO
   40 REM ELBUG   MAY 1986
   50 REM PROGRAM SUBJECT TO COPYRIGHT
   60 :
   70 ON ERROR GOTO 1540
   80 :
  100 H%=0
  110 MODE2:DIM maze%(18,26)
  120 FORA%=8TO15:VDU19,A%,A%-7,0,0,0:NE
XT
  130 *FX9,1
  140 *FX10,1
  150 na$="NOBODY"
  160 VDU23,1,0;0;0;0;
  170 CLS
  180 COLOUR2:PRINTTAB(6,3)"PAINTER"
  190 sc$="High score is "+STR$(H%)
  200 COLOUR3:PRINTTAB((20-LENsc$)/2,8);
sc$
  210 sc$="held by "+na$
  220 PRINTTAB((20-LENsc$)/2,10);sc$
  230 COLOUR6:PRINT''" Space Bar to star
t";
  240 REPEAT UNTIL GET=32
  250 CLS
  260 VDU23,224,&18,&18,&7E,&5A,&18,&24,
&24,&24
  270 VDU23,225,&81,&66,&24,&7E,&FF,&FF,
&66,&24
  280 VDU23,255,&FF,&FF,&FF,&FF,&FF,&FF,
&FF,&FF
  290 S%=0:L%=3:B%=0
  300 REPEAT
  310 COLOUR128:B%=B%+1:CLS:PROCBOARD(B%
):COLOUR3:PRINTTAB(7,0)"PAINT!";
  320 COLOUR128
  330 COLOUR4:PRINTTAB(1,28)"SCORE:";:CO
LOUR135:PRINT"000000";:PROCscore:COLOUR1
28
  340 PRINT" LI:";:COLOUR135:PRINT"0";L%
:COLOUR128:IF(B%+4)<>7 COLOURB%+4 ELSECO
LOURB%+5
  350 PRINTTAB(1,30)"TIME:"SPC8"MZ:";:PR
INT"0";B%:COLOUR128
  360 GCOL0,6:MOVE832,64:MOVE448,64:PLOT
85,832,32:PLOT85,448,32
  370 X%=2:Y%=2:COLOUR4
  380 IFL%>0 REPEATU%=RND(18):V%=RND(10)
+12:UNTILmaze%(U%,V%)<>1
  390 PRINTTAB(X%,Y%);CHR$224
  400 PRINTTAB(U%,V%);CHR$225
  410 TIME=0
  420 dead%=FALSE:check%=FALSE
  430 REPEAT
  440 PROCtime
  450 COLOUR135:COLOUR4
  460 IFU%>X% ANDmaze%(U%-1,V%)<>1 PRINT
TAB(U%,V%);CHR$32:maze%(U%,V%)=2:U%=U%-1
:PRINTTAB(U%,V%);CHR$225
  470 IFU%<X% ANDmaze%(U%+1,V%)<>1 PRINT
TAB(U%,V%);CHR$32:maze%(U%,V%)=2:U%=U%+1
:PRINTTAB(U%,V%);CHR$225
  480 IFV%>Y% ANDmaze%(U%,V%-1)<>1 PRINT
TAB(U%,V%);CHR$32:maze%(U%,V%)=2:V%=V%-1
:PRINTTAB(U%,V%);CHR$225
  490 IFV%<Y% ANDmaze%(U%,V%+1)<>1 PRINT
TAB(U%,V%);CHR$32:maze%(U%,V%)=2:V%=V%+1
:PRINTTAB(U%,V%);CHR$225
  500 IFU%=X% ANDV%=Y% PROCloselife
  510 IFT%>375 PROCloselife:TIME=0:GCOL0
,6:MOVE832,64:MOVE448,64:PLOT85,832,32:P
LOT85,448,32
  520 IFINKEY(-98) ANDmaze%(X%-1,Y%)<>1:
SOUND1,1,X%*10,2:k%=maze%(X%,Y%):PRINTTA
B(X%,Y%);CHR$32:maze%(X%,Y%)=2:X%=X%-1:P
RINTTAB(X%,Y%);CHR$224:IFk%=0 S%=S%+10*B
%:PROCscore
  530 IFINKEY(-67) ANDmaze%(X%+1,Y%)<>1:
SOUND1,1,X%*10,2:k%=maze%(X%,Y%):PRINTTA
B(X%,Y%);CHR$32:maze%(X%,Y%)=2:X%=X%+1:P
RINTTAB(X%,Y%);CHR$224:IFk%=0 S%=S%+10*B
%:PROCscore
  540 IFINKEY(-73) ANDmaze%(X%,Y%-1)<>1:
SOUND1,1,Y%*10,2:k%=maze%(X%,Y%):PRINTTA
B(X%,Y%);CHR$32:maze%(X%,Y%)=2:Y%=Y%-1:P
RINTTAB(X%,Y%);CHR$224:IFk%=0 S%=S%+10*B
%:PROCscore
  550  IFINKEY(-105) ANDmaze%(X%,Y%+1)<>
1:SOUND1,1,Y%*10,2:k%=maze%(X%,Y%):PRINT
TAB(X%,Y%);CHR$32:maze%(X%,Y%)=2:Y%=Y%+1
:PRINTTAB(X%,Y%);CHR$224:IFk%=0 S%=S%+10
*B%:PROCscore
  560 IFINKEY(-74) ANDX%=2 ANDY%=2 PROCc
heck
  570 UNTILdead%=TRUE OR cheok%=TRUE
  580 IF L%<>0 AND NOT check% THEN 370
  590 UNTILL%=0 ORB%=5
  600 IF B%=5 B%=0:S%=S%+1000:PROCscore:
GOTO300
  610 COLOUR128
  620 PRINTTAB(5,13)"GAME OVER!"
  630 T%=TIME+400:REPEAT UNTIL TIME>T%
  640 CLS
  650 IFS%>H% H%=S%:PROCinputname
  660 PRINT'"PRESS A KEY";: IF GET:GOTO1
60
 1000 DEFPROCinputname
 1010 COLOUR2:PRINT'"Well done!"
 1020 COLOUR3:PRINT'"You have the high"'
"score"
```

```
 1030 COLOUR5:PRINT'"Please enter your"'
"name."
 1040 COLOUR6:*FX15
 1050 PRINT:INPUTna$:ENDPROC
 1060 DEFPROCscore
 1070 COLOUR4:PRINTTAB(13-LEN(STR$(S%)),
28);S%;:ENDPROC
 1080 DEFPROCloselife:L%=L%-1
 1090 dead%=TRUE:SOUND0,-15,20,10
 1100 VDU19,0,8;0;
 1110 FOR I%=0 TO 3000:NEXT
 1120 COLOUR B%+3:IF B%+5=7 COLOUR B%+4
 1130  PRINTTAB(X%,Y%)CHR$224:PRINTTAB(U
%,V%)CHR$32
 1140 PRINTTAB(19-LEN(STR$(L%)),28);L%
 1150 VDU19,0,0;0;:ENDPROC
 1160 DEFPROCcheck
 1170 COLOUR128:PRINTTAB(6,0)"CHECKING"
 1180 check%=TRUE
 1190 FORK%=1TO25:FORR%=1TO18:IFmaze%(R%
,K%)=0 check%=FALSE
 1200 NEXT,
 1210 PRINTTAB(6,0)" PAINT! "
 1220 ENDPROC
 1230 DEFPROCtime:GCOL0,4:T%=TIME DIV35:
MOVET%+448,32:DRAWT%+448,64:ENDPROC
 1240 DEFPROCBOARD(B%)
 1250 ENVELOPE1,1,34,56,89,129,178,192,1
26,126,-126,-126,126,126
 1260 RESTORE(1340+(B%-1)*40)
 1270 IFB%<>7 COLOURB% ELSECOLOURB%+1
 1280 FORY%=1TO25
 1290 READA$
 1300 FORX%=1TO18:maze%(X%,Y%)=VAL(MID$(
A$,X%,1))
 1310 IFmaze%(X%,Y%)=1 VDU31,X%,Y%,255
 1320 NEXT,
 1330 ENDPROC
 1340 REM MAZE1 DATA
 1350 DATA111111111111111111, 1000000000
00000001, 1011101110111111101, 1000000000
00000001, 1110111011101110111, 1000000000
00001011, 1010111111111101011, 1010100000
00101011, 1010001011100010001, 1010111000
11111101, 1000000010000000001
 1360 DATA111110111111111101, 1000000010
00000001, 1011101010111101111, 1001101010
00000001, 1101101011010101101, 1001101000
00000001, 1011001011111111111, 1011011000
00000001, 1010011011011111101, 1010110001
00000001, 1010100101010110111
 1370 DATA1000001100010010111, 1111111111
11100011, 1111111111111111111
 1380 REM MAZE 2 DATA
 1390 DATA111111111111111111, 1000000000
00111111, 1010111101100000001, 1010011100
11101101, 1011011111010000101, 1001001110
10110101, 1101100110000010001, 1000110110
11011111, 1010000000011000001, 1000101010
01010101, 1111010100100010001
 1400 DATA1000001100110000101, 1010100110
01110001, 1000110111011111101, 1101110111
00000101, 1000000001011110101, 1011111101
01110101, 1000000000000000101, 1111111111
10110101, 1100000000000010101, 1001010101
01000101, 1011000000001110101
 1410 DATA1001010101010000101, 1100000000
00010001, 1111111111111111111
 1420 REM MAZE 3 DATA
 1430 DATA111111111111111111, 1000000001
11110001, 1011111010000110101, 1010000101
10010001, 1010111010111000111, 1010110101
00010111, 1000000100010000011, 1111111111
01101011, 1000000010001000011, 1010101000
10001011, 1000001110110110001
 1440 DATA111111110100010101, 1110000001
01010001, 1100100010100011111, 1001001001
11111111, 1010010000000000011, 1000100111
11111011, 1001001000011111001, 1010010010
01111101, 1010110111100001101, 1010010010
01101101, 1011001000110000101
 1450 DATA1000101111111010101, 1110000000
00000001, 1111111111111111111
 1460 REM MAZE 4 DATA
 1470 DATA111111111111111111, 1000000000
10000001, 1011110110001011101, 1010000000
10000001, 1010101110001011101, 1000100000
10000001, 1101111011110111111, 1000001011
00000001, 1010100000010101101, 1010111111
01010101, 1010001110000010101
 1480 DATA1011100110110101101, 1011110010
11010101, 1000011000110101101, 1101001110
00010001, 1000100111111111011, 1010110011
10001011, 1000001000001010111, 1010101111
10101011, 1000000010000000011, 1010101010
10101011, 1000000010000000011
 1490 DATA1010101011111111111, 1000000000
00000001, 1111111111111111111
 1500 REM MAZE 5 DATA
 1510 DATA111111111111111111, 1000001000
00010001, 1010101011010101101, 1000001000
00010101, 1101011010110101101, 1000000000
00010001, 1010101101111110101, 1000000100
00010001, 1101010101110101101, 1000000100
11010101, 1111111110010101101
 1520 DATA1000001111000101101, 1010100000
10110101, 1010001110000000101, 1010100000
10110001, 1000001111000101101, 1111111110
01010101, 1110001100100101101, 1100111001
00110001, 1101110010011101101, 1000000100
11110101, 1011111001101101101
 1530 DATA1011111011101101101, 1000000000
00000001, 1111111111111111111
 1540 ON ERROR OFF
 1550 MODE 6
 1560 IF ERR<>17:REPORT:PRINT " at line
";ERL
 1570 END
```

Your Guide to BEEBUG

No alterations are required to the programs appropriate to the Electron from this month's BEEBUG. In fact, most of the programs in BEEBUG this month will work on an Electron. A few general comments on converting Beeb programs to the Electron may also prove useful.

Beeb programs sometimes fail to run properly on the Electron because of the Beeb's mode 7 and extra sound facilities. However, conversion rarely proves very difficult.

The Beeb's mode 7 display is of the same format as the mode 6 display that ELBUG members know but with a full range of colours and crude graphics. Mode 7 is not available on the Electron. However, the Electron defaults to mode 6 when it encounters a MODE 7 command. So BBC micro programs that include a mode 7 display will therefore run on the Electron but often with a corrupted (and monochrome) display.

The Beeb has three sound channels each with full control over the volume of notes. Electron Basic includes all the SOUND and ENVELOPE parameters that are used in the BBC micro even though several have no effect. This means that Beeb programs making full use of sound will still run on the Electron unaltered, though not, of course, producing the full sound effect.

The other major difference between the Electron and the BBC micro is speed. The Electron is much slower, and particularly slow in modes 0, 1, and 2. BBC micro programs often benefit from a change of mode (admittedly, with a loss of some colours) when running them on the Electron. When changes are needed they will be given in this ELBUG supplement.

MANDELBROT GRAPHICS

This amazing program creates an almost infinite range of fascinating and brightly coloured screen displays. It's unlikely to help your maths but it will keep you happy for hours.

The program is fully compatible with the Electron.

PROWORD AND WORDPOWER

ProWord is fully compatible with the Electron and provides a useful alternative to the established word processors for this machine - View (reviewed in ELBUG Vol.2 No.5) and StarWord (reviewed in Vol.3 No.2). Note, however, that you will only be able to use ProWord if you have a ROM board fitted to your Electron.

THE BARRY BOX

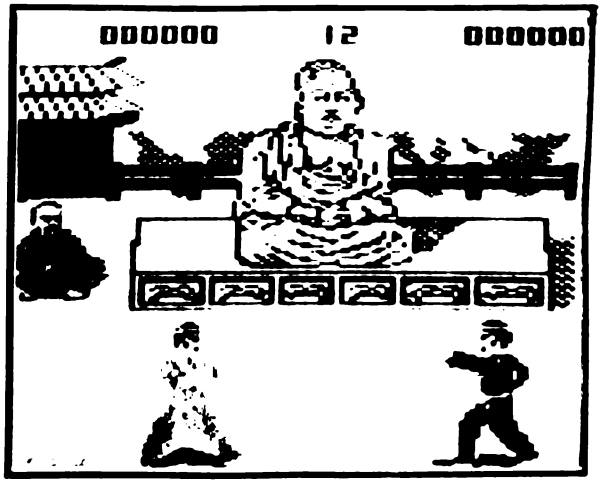
The Barry box uses the 1MHz bus on the BBC micro. There is no 1MHz bus interface yet available for the Electron, so this fun add-on is not suitable for Elk owners.

WINDOWS AND ICONS

This application program to use the USI utility will not operate correctly on the Electron. This is due to a very subtle use in the program of the 6845 graphics controller chip in the Beeb which the Electron does not contain.

MARTIAL ARTS

These three games are all doing very well in the charts at the moment, which is no surprise as they are all excellent. What's more, they are all available in Electron versions.



MAIL-MERGE WITH FILER

Because of the memory taken up by the ADFS in the Electron's Plus-3, neither the Filer program alone nor with this

extension will operate on the Electron. However, if you have a disc interface that does not use the ADFS (such as Cumana's Electron disc interface) then the program will operate correctly.

SOFT SCREEN SHUFFLE

This short utility to make the presentation of display screens more attractive is not compatible with the Electron because of its use of the 6845 display controller chip in the BBC micro. This chip is not included in the Electron.

DO-IT-YOURSELF BASIC (Part 3)

In this second article Alan Webster adds more commands to the interpreter covered last month.

The program works perfectly on the Electron

PRINTER SURVEY

ELBUG members with a Plus-1 or other printer interface will be as interested as Beeb owners in the latest printers on the market.

If you are thinking of buying a printer, check it out here first.

*SPOOL GRAPHICS

Any technique for speeding up your graphics programs and fitting them in less memory has got to be useful. That is just what this utility does, and it all works on the Electron.

WORKSHOP

Stacks and Queues

The two routines described will work just as well on the Electron, so Elk owners can also learn about these useful techniques. The demo program referred to in BEEBUG makes extensive use of mode 7, so the ELBUG cassette just contains the relevant procedures.

SIDEWAYS RAM MODULES

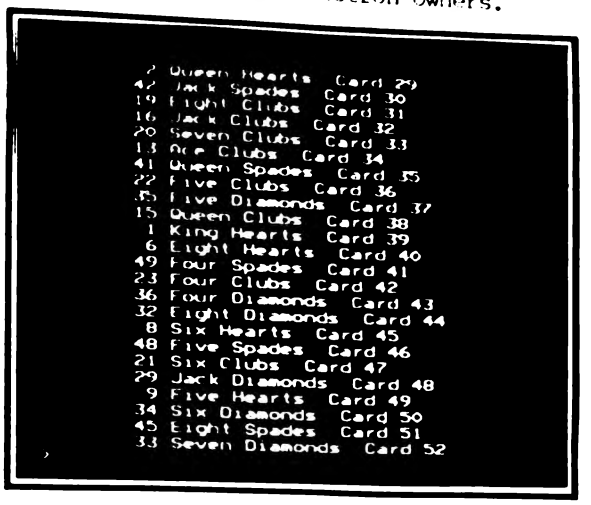
It is not possible to fit sideways RAM modules of this kind to the Electron. However, ELBUG members have already met a sideways RAM module - from ACP and reviewed in ELBUG in Vol.2 No.10.

FIRST COURSE

Using Random Numbers

Using random numbers on a micro is a seemingly simple subject, but there's more to it than you'd think. This beginner's article will help you to understand this useful feature of Electron Basic.

Everything that is contained in this article is useful to Electron owners.


Queen Hearts Card 29
Jack Spades Card 30
Eight Clubs Card 31
Jack Clubs Card 32
Seven Clubs Card 33
Ace Clubs Card 34
Queen Spades Card 35
Five Clubs Card 36
Five Diamonds Card 37
Queen Clubs Card 38
King Hearts Card 39
Eight Hearts Card 40
Four Spades Card 41
Four Clubs Card 42
Four Diamonds Card 43
Eight Diamonds Card 44
Six Hearts Card 45
Five Spades Card 46
Six Clubs Card 47
Jack Diamonds Card 48
Five Hearts Card 49
Six Diamonds Card 50
Eight Spades Card 51
Seven Diamonds Card 52

ADVENTURE GAMES

Neither the game nor the book reviewed this month are compatible with the Electron.

HINTS

Of this month's hints and tips, the following are applicable to Electron owners:

MERGING VIEWSTORE FILES
PRINTER CHECK
INITIALIZING A DATA AREA
DATE CHECKING

ACROSS THE TUBE

Although Acorn has demonstrated Tube interfaces running on an Electron, no production model has ever been produced so this complex subject will not be of interest to many ELBUG members.

EBONY CASTLE

Can you survive in Ebony Castle? This excellent game will test your game-playing abilities to the full.

What's more, it's fully compatible with the Electron.

FIND AND REPLACE

Jagdish Sah describes a pair of useful routines that will provide a useful aid when developing programs.

This program will add two extra commands to your Electron's Basic. FIND searches a Basic program in the memory for a specified string and lists all lines in which it is present. REPLACE replaces occurrences of a specified string with another. The strings can be variable names, Basic keywords, or just a series of characters. I shall describe briefly how the program works later but first let's see how to use it.

Type in the program and save it on tape or disc before you run it. Then save the machine code routine produced with the help of the displayed command. When you intend to use the utilities, type:

```
*RUN FND/REP
```

This will install both utilities. Now to find a string in your program, type:

```
FIND <string>
```

where <string> is the string to be searched. You must include the space following FIND. Everything after this space is treated as the part of the string itself. Therefore, several words or Basic keywords with or without preceding, embedded and trailing spaces may be used as the search string. The routine itself does not impose any limit on the length of the string. But the operating system restricts the total length of a typed line to 238 characters. You can try it out with this program itself. Type:

```
FIND EQU
```

This will list all lines which contain EQU, i.e., both EQUB and EQUS. The first occurrence of the search string in the line will be highlighted in inverse video. If the string is present more than once in a line its second and subsequent occurrences will not be highlighted.

The REPLACE utility can be used in selective and global modes. In the selective mode its syntax is:

```
REPLACE <oldstring>,<newstring>
```

Again the space must be included, as must the comma separating the strings.

The routine searches the Basic program in memory for the presence of the old string. If it is found, a formatted listing of the whole line is displayed with the string to be replaced highlighted as before. Now you are asked whether to continue with the substitution. It would be carried out only if the response is 'Y'. Otherwise the old string is left unchanged. This process is repeated until the end of the program is reached. If the old string is present more than once in a single line, the routine will display the same line for each occurrence but it will highlight only the specific occurrence intended for substitution. If Escape is pressed at any time the routine is terminated immediately.

The global form has the syntax:

```
REPLACEG <oldstring>,<newstring>
```

The G must follow immediately after REPLACE without a space in between. The rest of the command is similar to that in the selective mode. In this case the routine searches the entire program for the presence of the old string replacing each occurrence with the new string without displaying any messages.

Try out REPLACE with:

```
REPLACE CPY,COPY
```

Answer all queries with 'Y'. Now list the program to check whether it worked properly. If you want to restore the string, use the command

```
REPLACEG COPY,CPY
```

A comma is used to separate the two strings. Sometimes strings containing commas may be required to be replaced. For this reason, the separator can be easily altered. You can either change line 1170 of the program (to, say, '@' instead of ',') and re-assemble the code or, with the machine code routine in memory, enter:

```
?&8C0=ASC("@")
```

Now '@' becomes the separator.

Note that the utilities work only as immediate mode commands - they cannot be included as statements in a program.

PROGRAM NOTES

If the Break key is pressed at any time after loading the utilities they will no longer work. To re-install them, type:

CALL &8C1

The machine code routine takes just over two pages of memory and &8C0 is used as its starting address. However, this occupies the area (&8C0-8FF) where envelope definitions are stored. If you find this inconvenient, alter the starting address by modifying line 100.

HOW IT WORKS

Let us consider the line:

FIND PRINT

As soon as it is typed in, the Basic interpreter tokenizes as much of it as it can and puts the whole line in memory at &700. Next it tries to execute it. Immediately it realizes that the term 'FIND' is unknown to it. So it tries to print an error message. An indirect jump is made through the Break vector located at &202. This routine intercepts this indirect jump and checks whether the first word typed was FIND or REPLACE. If not, it returns the control to the operating system to print the error message. Otherwise it fetches the rest of the line, in this case the tokenized form of PRINT, and searches for its occurrence in the current program in memory. If it finds the string, the whole line is listed employing routines present in the Basic ROM.

The same explanation also applies to the REPLACE utility. However, it also replaces the old string with the new one. Again the program uses the Basic ROM routines to substitute one string with another.

```
  10 REM PROGRAM FIND and REPLACE
  20 REM VERSION 1.1
  30 REM AUTHOR  Jagdish Sah
  40 REM ELBUG   MAY 1986
  50 REM PROGRAM SUBJECT TO COPYRIGHT
  60 :
 100 start=&8C0
 110 PROCassemble
 120 PRINT'"To save the machine code pr
oduced type"'
 130 PRINT" *SAVE FND/REP ";~start;" ";
~P%;" ";~(start+1)'
 140 END
 150 :
1000 DEF PROCassemble
1010 page=&18:lineptr=&B:intA=&2A
1020 strptr=&70:len=&72:len2=&73
1030 linelen=&74:ystr=&75:ystr2=&76
1040 yline=&77:cmdflg=&78
1050 temp=&79:qflag=temp+1
1060 delimiter=&7B:yfound=&7C
1070 strbuff=&600
1080 osasci=&FFE3:osnewl=&FFE7
1090 osbyte=&FFF4:osrdch=&FFE0
1100 READ char,keyword,lineno,chkprog
1110 READ goto,basic,insline
1120 REM cmdflg 0-FIND, 1-REPLACE, &FF-
REPLACEG
1130 :
1140 FOR pass=0 TO 3 STEP 3
1150 P%=start:[ OPT pass
1160 .separator
1170 EQUB(ASC(","))
1180 LDA&203:CMP#newrtn DIV 256:BEQ out
1190 STA oldvec+2
1200 LDA &202:STA oldvec+1:SEI
1210 LDA #newrtn MOD 256:STA &202
1220 LDA #newrtn DIV 256:STA &203:CLI
1230 .out RTS
1240 :
1250 .findlen STY ystr2
1260 .getstr LDA (strptr),Y:INY
1270 BEQ oldrtn2
1280 CMP delimiter:BNE getstr
1290 DEY:TYA:SEC:SBC ystr2:RTS
1300 :
1310 .oldrtn2 PLA:PLA
1320 .oldrtn PLA:TAY:PLA:TAX
1330 LDA &FC
1340 .oldvec JMP 0
1350 :
1360 .newrtn TXA:PHA:TYA:PHA
1370 JSR chkprog
1380 LDA #&0D:STA delimiter
1390 LDY #0:STY cmdflg
1400 .chkcmd
1410 LDA replcmd,Y:BEQ foundrepl
1420 CMP &700,Y:BNE norepl
1430 INY:BNE chkcmd
1440 .norepl LDY #0
1450 .chkfind
1460 LDA findcmd,Y:BEQ cmdok
1470 CMP &700,Y:BNE oldrtn
1480 INY:BNE chkfind
1490 .foundrepl
1500 LDX separator:STX delimiter
1510 INC cmdflg
1520 LDA &700,Y:CMP #ASC("G")
1530 BNE cmdok:INY
1540 LDX #&FF:STX cmdflg
1550 .cmdok
1560 LDA &700,Y:CMP#ASC(" "):BNE oldrtn
```

```
1570 INY:STY strptr
1580 LDA #strbuff DIV 256:STA strptr+1
1590 PLA:PLA:LDY #0
1600 .savestr
1610 LDA &700,Y:STA strbuff,Y
1620 INY:BNEsavestr:STY&700:STY lineptr
1630 LDA page:STA lineptr+1:JSR findlen
1640 STA len:LDA cmdflg:BEQ mainloop
1650 LDA #&0D:STA delimiter
1660 INY:JSR findlen:STA len2
1670 :
1680 .mainloop
1690 LDA #0:STA &D3
1700 LDY #4:STY yline:JSR setlineno
1710 BPL notfinished
1720 .exit JMP basic
1730 .notfinished
1740 DEY:STY ystr
1750 .compare
1760 LDX &FF:BMI exit:LDY ystr:LDA (str
ptr),Y
1770 INC ystr:STA temp:LDY yline
1780 LDA (lineptr),Y:INC yline
1790 CMP temp:BNE mismatch
1800 LDA ystr:CMP len:BNE chklinelen
1810 JSR listline
1820 LDAcmdflg:BEQnextline:JSR exchange
1830 JSR setlineno
1840 .mismatch
1850 DEC ystr:BEQ ylok:DEC yline
1860 .ylok LDA #0:STA ystr
1870 .chklinelen
1880 LDY yline:CPY linelen:BNE compare
1890 .nextline
1900 CLC:LDA linelen:ADC lineptr
1910 STA lineptr:BCC mainloop
1920 INC lineptr+1:BNE mainloop
1930 :
1940 .setlineno
1950 LDY #3:LDA (lineptr),Y:STA linelen
1960 DEY:LDA (lineptr),Y:STA intA:DEY
1970 LDA (lineptr),Y:STA intA+1
1980 .exitlist RTS
1990 :
2000 .listline
2010 LDA yline:SEC:SBC len:STA yfound
2020 LDX cmdflg:BMIexitlist:BEQskipnewl
2030 JSR osnewl
2040 .skipnewl
2050 LDY #0:STY qflag:JSR lineno+4
2060 LDA #&20:JSR osasci:LDY #4
2070 .printloop
2080 CPY yfound:BNE notyf
2090 LDA #255:STA &D3:JMP notyl
2100 .notyf
2110 CPY yline:BNE notyl:LDA #0:STA &D3
2120 JSR osasci
2130 .notyl
2140 LDA (lineptr),Y:CMP #ASC("""")
2150 BNE notquote:EOR qflag:STA qflag
2160 LDA #ASC("""")
2170 .notquote
2180 LDX qflag:BNE inquotes
2190 LDX cmdflg:BEQ notcolon
2200 CMP #58:BNE notcolon:JSR osnewl
2210 LDX #5:LDA #&20
2220 .colon
2230 JSR osasci:DEX:BNE colon
2240 LDA #58
2250 .notcolon
2260 JSR goto:BCC notgoto:STY temp
2270 JSR lineno:LDY temp:BNE chkl
2280 .notgoto
2290 JSR keyword:CLC:BCC chk
2300 .inquotes JSR char
2310 .chk INY
2320 .chkl
2330 CPY linelen:BNEprintloop:JMPosnewl
2350 .exchange
2360 LDX cmdflg:BMI yes2:DEX
2370 .printmes
2380 LDA mes,X:BEQ mesdone:JSR osasci
2390 INX:BNE printmes
2400 .mesdone
2410 LDA &FF:BMI escape:LDA #15
2420 LDX #1:JSR osbyte:JSR osrdch
2430 CMP #&1B:BEQ escape:AND #&5F
2440 CMP #ASC("Y"):BEQ yes
2450 CMP #ASC("N"):BNE mesdone
2460 JSR osasci:JSR osnewl
2470 .escape RTS
2480 .yes JSR osasci:JSR osnewl
2490 .yes2 JSR setlineno
2500 LDA yfound:STA temp:LDY #4
2510 .move LDA (lineptr),Y
2520 STA &700,Y:INY:CPY temp:BNE move
2530 LDA ystr2:STA ystr
2540 .exch
2550 LDY ystr:LDA (strptr),Y
2560 CMP #&0D:BEQ move2:INC ystr
2570 LDY temp:STA &700,Y:INC temp
2580 BNE exch
2590 .move2
2600 LDY yline:INCyline:LDA(lineptr),Y
2610 LDY temp:INC temp:STA &700,Y
2620 CMP #&0D:BNE move2:LDA yfound
2630 CLC:ADC len2:STA yline
2640 LDY #4:JMP insline
2650 :
2660 .mes EQUB(&0D)
2670 EQUS("Replace (Y/N)? "):EQUB(0)
2680 .replcmd
2690 EQUS("REPLACE"):EQUB(0)
2700 .findcmd
2710 EQUS("FIND"):EQUB(0)
2720 ]:NEXT:ENDPROC
2730 :
2740 REM Basic entry points
2750 DATA &B558,&B50E,&991F,&BE6F
2760 DATA &97E7,&8AF6,&BC8D
```